Unsupervised Clustering: Speed, accuracy, and machine precision

Nathaniel Beckemeyer

1 Algorithm Introduction

Given several artificially generated datasets (and one dataset of our own choosing), we tested and compared the accuracy and performance of two clustering algorithms, k-means and the expectation-maximization (EM) algorithm for a mixture of Gaussians.

1.1 K-means

K-means is a simple clustering algorithm. Given a set of k points to serve as the starting means, it iteratively assigns points to their closest mean (as determined by some distance metric). Then, the algorithm recalculates the means as the average of the locations of the points assigned to each of them. This process repeats until the means no longer change.

1.2 EM for a Mixture of Gaussians

The EM algorithm consists of two steps: An expectation step, and a maximization step. It assumes that there are latent variables responsible for the distribution of the data, and that the data accurately reflects these variables. For this project, our hidden variables were the means, standard deviations, and prior probabilities of being generated by a cluser (ϕ_j, henceforth) of the k clusters that 'generated' (or are generated by) our dataset.

The expectation step is a measure of how well our latent variables (means, standard deviations, and Φ) predict the actual dataset: A vector of weights is assigned to each of the k clusters to indicate how well the corresponding set of latent variables predict the dataset; the length of the weight vector is therefore equal to the number of training examples. This weight vector is then normalized across all of the clusters.

The maximization step is then to recalculate the latent variables so as to 'maximize' the likelihood that the dataset would occur given the expectations.

Mathematically, if we allow z_j to represent the latent variables for cluster j, x_i to represent the ith example in the dataset, and w_{ji} to represent the

weight (expectation) of the ith example of the jth vector of weights, then the expectation step is

$$w_{ji} \leftarrow p(x_i = j|z_j) \tag{1}$$

and the maximization step is then

$$z_j \leftarrow p(X|z_j, w_j)p(z_j) \tag{2}$$

Note that $p(z_j)$ is just our prior for cluster j, and that after the weights are normalized,

$$p(z_j) = mean(w_j) = \phi_j$$

Then, for a mixture of Gaussian data, where μ_j and Σ_j represent the mean and covariance matrix, respectively, of the jth Gaussian distribution (which has as many dimensions as features in each input example), the expectation step is

$$w_{ji} \leftarrow \alpha \,\mathrm{mvnpdf}(X; \mu_j, \Sigma_j)\phi_j \tag{3}$$

where mvnpdf is the probability density function for a multivariate normal distribution and α is a normalizing factor.

The maximization step is

$$\mu_j \leftarrow \bar{X}_{w_j} \tag{4}$$

$$\Sigma_j \leftarrow \hat{\Sigma}_{w_j} \tag{5}$$

where $\bar{X}_{w_j}$ is the average of the dataset weighted by w_j, and $\hat{\Sigma}_{w_j}$ is the weighted sample covariance.

This iterative process repeats until the latent variables no longer change.

2 Implementation Saga

Implementing k-means was straightforward. I initialized the cluster means by sampling without replacement from the dataset (to ensure that all clusters have at least one datapoint assigned to them). Then, I proceeded with the algorithm. The language that I used to implement this algorithm was Python. The measurement of convergence was equality to machine precision (as opposed to a user-specified tolerance).

The EM algorithm proved much more stressful (and difficult to work with). I implemented it first in GNU Octave, a Matlab look-alike. It worked on the small datasets, but I could not get it to terminate on the larger datasets. So, I decided to implement the algorithm in Java because I had a matrix library ready, so the code could be as straightforward to implement as with Octave while gaining a small speed boost.

The speed boost was small. Too small, in fact, for that algorithm to terminate on the larger datasets on my computer.

Needing efficiency, I decided to implement the algorithm in C. But there was a strange problem, something that Sam Beckmann said that he and William

Macke had encountered while implementing their algorithms: The means would be startingly close to each other, suggesting that the algorithm did not terminate properly. I experienced the same problem. I suspected that the problem lay in the calclulation of the pdf function, which involved exponentiation, and eventually returned infinite values.

Nick Owen said the he had used NumPy with Python, and indicated that he was using the logarithm of the value to avoid this problem. I tried to do the same thing—but I could not figure out how he normalized the logarithms of the probabilities while maintaining the correct ratios in the exponential space (in other words, I could not find a function g that created a homomorphism from the normalization operation in exponential numbers to g in logarithmic numbers). I tried removing the logarithm, but alas, I ran into the same problem as with the C code.

Wondering if perhaps my programming skills were simply so subpar that I was ruining the algorithm by somehow misuisng side effects (after all, it worked perfectly in Octave), I chose a language where side effects must be declared explicitly: Haskell. This implementation worked with the small datasets, but ran into the same problem that I had with the C and Python code. I investigated this issue further, and realized that the means converged slowly.

The convergence test that I was using tested the current means to the means from the previous iteration; if less than some speciifed threshold, the algorithm would terminate. The threshold that I was choosing was apparently small enough for the smaller datasets, but too large for the larger datasets. So, I decided to decrease the tolerance.

Alas, it was too slow.

I decided to download a stranger's implementation of the EM algorithm as a base for comparison, or to help debug my own.

It was plagued by the same problems that my implementation was.

Knowing that my implementation worked when I used matrix libraries, and also happened to be too slow, I decided to use C++ with a matrix library—Armadillo in particular. It was fast enough, but I was getting similar results to my C implemenation; as always, however, my unit testing indicated that there was nothing wrong, Fortunately, however, C++ started returning infinite values, as well. I realized that I was dividing by 0.

The matrix method of implementing mvnpdf is efficient, nonintuitive, and easy to debug—unlike the typical imperative implementation. The first step is to perform a Cholseky decomposition of the covariance matrix. In the final calculation of the probability density, a value is divided by the product of the diagonal of this decomposition. That value was 0, so I removed it to be swept up by the later normalization step.

But of course, this factor was ultimately calculated from Σ_j, so it could not be logically removed—a fact reflected in the nonsensicality of the results of my now-terminating algorithm.

To solve this problem, I set a minimum threshold of $1E{-}10$. Then, the algorithm terminated properly. In the C++ implementation, I tried to initialize the algorithm by randomizing the expectation step and by randomizing the

maximization step. The algorithm was almost an order of magnitude faster when I randomized the maximization step first, so that's what I used.

But some datasets were still slow—the algorithm would terminate within a reasonable timeframe, but too long to run repeated trials. Jon Bolin helped me get the software ready to run on the Tandy Supercomputer. With all 32 of its cores running, we tried to see if it would be fast.

It was slower than my laptop. So, I ran these tests for more than 60 hours nonstop on my computer. Each artificial trial ran only 10 times in that timespan, so I'm only using 10 runs for those results.

3 Datasets

3.1 Artificial Datasets

The artifical datasets fell into four segments, and test cases were peformed on each of them. A complete listing of the parameters of each test case can be found in Table 1. Each test case was run 10 times, unless noted otherwise, to create the results presented in Section 4. Each data point generated by these generators had one feature: A real number.

3.2 Real Life Dataset

The dataset that I chose for my real life dataset was the 3D spatial network dataset from the UCI repository. For preprocessing, I cut the dataset down to its first column; though my EM algorithm could readily handle the multiple features, my k-means algorithm did not have an implementation of the mvnpdf function. Then, for speed, I chose a random sample of 25000 datapoints. I ran 50 trials for this dataset.

4 Results

4.1 Artifical results

In order to evaluate my results on the artifical datasets, I calculated the likelihood that two points chosen to be in the same cluster by EM or k-means was generated by the same source, which was known.

For each test case, we report 10 results:

1. The mean means for k-means
2. The mean standard deviation for the above means
3. The mean number of trials before k-means converged
4. The standard deviation of the number of trials before convergence
5. The mean means for EM
6. The mean standard deviation for the EM means

Tab. 1: Test cases. N represents the number of point per source, σ represents the standard deviation of each generator, S represents the number of sources, R represents the spacing between each source, and k is the number of clusters formed by the algorithms.

Trial	Part	N	σ	S	R	k
1	1	1000	1	3	0.5	2
2	1	1000	1	3	0.5	3
3	1	1000	1	3	0.5	6
4	1	1000	1	3	0.5	8
5	1	1000	1	3	1	2
6	1	1000	1	3	1	3
7	1	1000	1	3	1	6
8	1	1000	1	3	1	8
9	1	1000	1	3	1.5	2
10	1	1000	1	3	1.5	3
11	1	1000	1	3	1.5	6
12	1	1000	1	3	1.5	8
13	1	1000	1	3	2	2
14	1	1000	1	3	2	3
15	1	1000	1	3	2	6
16	1	1000	1	3	2	8
17	1	1000	1	5	0.5	5
18	1	1000	1	5	1	5
19	1	1000	1	5	1.5	5
20	1	1000	1	5	2	5
21	1	1000	1	10	0.5	10
22	1	1000	1	10	1	10
23	1	1000	1	10	1.5	10
24	1	1000	1	10	2	10
25	2	1000	1	5	3	5
26	2	1000	2	5	3	5
27	2	1000	3	5	3	5
28	3	1000	$\sim U[0.75, 2]$	5	1.25	5
29	4	100	0.75	5	1.25	5
30	4	1000	0.75	5	1.25	5
31	4	10000	0.75	5	1.25	5

7. The mean number of trials before EM converged
8. The standard deviation of the number of trials before convergence
9. The accuracy of k-means
10. The accuracy of EM

See table 2 for the summary of the artificial test results. For more specific results, see Appendix A.

EM and k-means performed remarkably similarly on these artificial trials, but EM outperformed k-means on 29 of the 31 trials. Considering that only 10 trials could be performed, I'm unwilling to draw further conclusions about which algorithm performs better on this Gaussian data, in general. Though, I did expect EM to outperform k-means quite substantially, given the fact that the assumptions underlying out EM implementation align with the data generators. I suspect that further analysis would only tip the scale further in favor of EM.

4.2 Real-life dataset results

For the real-life dataset, however, I instead calculated the average probability of observing each point, assuming that it belonged to the cluster to which it was assigned for k-means, or the maximum likelihood for EM. Naturally, I expect EM to perform better at this task, as it is part of what EM does to calculate the clusters to begin with—of course, I would expect EM to perform better on a mixture of Gaussians to begin with because that is explicitly the purpose of our implementation.

And as I predicted, EM greatly outperformed k-means in this task. Due to underflow in the k-means value, I switched to use logarithms. EM had an average point observation logarithm probability of -17.99. The same value for k-means was $-7.13\text{E}27$. Clearly, EM greatly outperformed k-means in this area.

5 Conclusions

Implementing these algorithms was interesting. I learned a lot about clustering, and I feel that I understand both, but EM in particular, extremely well. I have never worked on a project that was quite as frustrating or confusing, given the numerical stability issues. I did, however, learn a very important lesson about the usage of logarithmic math from the beginning. I also learned that my implementation of EM (which I thought would be fast) is actually extremely slow, and that I should probably use another implemntation. (I should also ensure that that implementation is thoroughly tested, as the implementation that I found online did not work!)

I also learned that k-means is a decent algorithm for clustering, only being slightly outperformed by EM; though, that lesson was not well-reflected in the real-life dataset. I am very glad to be done with this project, though I did learn much about how clustering works (and especially how it works in multiple dimensions).

Tab. 2: The simple artificial test results, from 10 runs. The bolded accuracy indicates the higher number.

	μ_n		σ_n		Accuracy (%)	
Trial	k-means	EM	k-means	EM	k-means	EM
1	11.0	16778.0	2.28	13791.77	**18.76**	18.75
2	29.4	22037.7	10.12	3852.19	13.19	**13.54**
3	33.5	81968.5	13.89	8024.25	7.28	**10.26**
4	53.4	85711.1	22.20	20386.4	5.71	**6.88**
5	10.5	14563.5	1.91	1792.84	21.85	**22.07**
6	18.9	100186.8	5.45	23655.33	15.71	**15.72**
7	41.0	146583.3	10.78	224463.55	8.50	**11.81**
8	41.0	210371.5	16.97	90369.85	6.63	**8.30**
9	10.5	2767.3	3.88	150.61	25.05	**25.32**
10	26.5	12869.4	11.08	3322.44	18.56	**19.26**
11	38.2	256251.3	11.31	134023.57	9.98	**12.04**
12	60.0	93188.1	35.92	26814.80	7.79	**9.07**
13	11.1	1189.2	0.70	375.00	26.63	**26.70**
14	17.4	4181.5	3.35	252.65	21.77	**21.80**
15	57.1	71038.5	14.69	32602.40	11.55	**13.97**
16	49.6	155632.0	17.37	74584.78	8.99	**10.04**
17	34.3	1410111.4	14.46	3286316.26	**5.79**	5.60
18	37.7	354688.1	7.98	53003.33	7.64	**7.84**
19	39.5	74615.9	12.25	6633.99	9.75	**10.36**
20	37.9	27741.1	11.24	8519.74	11.77	**12.06**
21	144.0	543234.0	33.06	143144.60	2.10	**2.45**
22	80.0	1862108.0	30.61	1285438.89	3.26	**3.33**
23	81.8	2256801.1	26.99	3056006.90	4.35	**4.49**
24	107.6	860958.3	44.30	514057.09	**5.49**	5.37
25	25.1	1674.1	8.87	200.2	15.76	**15.86**
26	37.3	134772.1	9.47	36496.79	9.97	**10.65**
27	56.8	74025.9	20.18	33300.77	7.61	**7.72**
28	48.0	78386.5	13.18	6287.79	7.96	**8.68**
29	15.7	9974.6	9.49	1078.83	10.69	**11.27**
30	53.5	46663.0	18.00	17215.69	10.30	**10.68**
31	89.1	125067.4	35.34	65548.70	10.43	**10.57**

A Extended results

These are the extended results for k-means and EM, containing the 4 values that represent each cluster.

Tab. 3: The specific artificial test results for k-means.

	Cluster									
Value	0	1	2	3	4	5	6	7	8	9
μ_{μ_1}	0.666	2.419								
σ_{μ_1}	0.001	0.001								
μ_{σ_1}	0.668	0.641								
σ_{σ_1}	0.000	0.000								
μ_{μ_2}	0.231	1.542	2.857							
σ_{μ_2}	0.012	0.020	0.020							
μ_{σ_2}	0.569	0.365	0.520							
σ_{σ_2}	0.002	0.003	0.005							
μ_{μ_3}	-0.539	0.478	1.213	1.903	2.632	3.524				
σ_{μ_3}	0.117	0.136	0.148	0.132	0.097	0.062				
μ_{σ_3}	0.469	0.241	0.203	0.203	0.228	0.400				
σ_{σ_3}	0.009	0.002	0.005	0.011	0.013	0.008				
μ_{μ_4}	-0.989	-0.067	0.602	1.173	1.703	2.247	2.876	3.692		
σ_{μ_4}	0.226	0.145	0.118	0.141	0.134	0.112	0.090	0.066		
μ_{σ_4}	0.440	0.215	0.172	0.160	0.151	0.166	0.202	0.387		
σ_{σ_4}	0.019	0.020	0.005	0.002	0.005	0.007	0.008	0.004		
μ_{μ_5}	0.962	3.051								
σ_{μ_5}	0.001	0.001								
μ_{σ_5}	0.761	0.753								
σ_{σ_5}	0.000	0.000								
μ_{μ_6}	0.352	1.901	3.494							
σ_{μ_6}	0.000	0.000	0.000							
μ_{σ_6}	0.596	0.447	0.621							
σ_{σ_6}	0.000	0.000	0.000							
μ_{μ_7}	-0.327	0.735	1.585	2.398	3.260	4.321				
σ_{μ_7}	0.137	0.169	0.159	0.157	0.130	0.087				
μ_{σ_7}	0.471	0.272	0.236	0.243	0.270	0.460				
σ_{σ_7}	0.018	0.007	0.008	0.006	0.011	0.011				
μ_{μ_8}	-0.664	0.291	1.003	1.625	2.248	2.901	3.622	4.566		
σ_{μ_8}	0.085	0.129	0.190	0.197	0.163	0.139	0.139	0.125		
μ_{σ_8}	0.425	0.228	0.188	0.175	0.182	0.195	0.237	0.436		
σ_{σ_8}	0.013	0.021	0.008	0.001	0.014	0.007	0.003	0.009		

μ_{μ_9}	1.201	3.777						
σ_{μ_9}	0.003	0.002						
μ_{σ_9}	0.872	0.902						
σ_{σ_9}	0.001	0.001						
$\mu_{\mu_{10}}$	0.714	2.594	4.424					
$\sigma_{\mu_{10}}$	0.002	0.002	0.000					
$\mu_{\sigma_{10}}$	0.696	0.521	0.690					
$\sigma_{\sigma_{10}}$	0.000	0.001	0.000					
$\mu_{\mu_{11}}$	-0.096	1.078	2.088	3.047	4.057	5.236		
$\sigma_{\mu_{11}}$	0.076	0.107	0.136	0.132	0.118	0.085		
$\mu_{\sigma_{11}}$	0.495	0.311	0.280	0.281	0.313	0.522		
$\sigma_{\sigma_{11}}$	0.013	0.005	0.003	0.003	0.009	0.010		
$\mu_{\mu_{12}}$	-0.483	0.509	1.331	2.107	2.857	3.634	4.473	5.517
$\sigma_{\mu_{12}}$	0.024	0.013	0.002	0.000	0.001	0.002	0.002	0.002
$\mu_{\sigma_{12}}$	0.434	0.248	0.218	0.212	0.219	0.226	0.265	0.497
$\sigma_{\sigma_{12}}$	0.004	0.006	0.002	0.000	0.000	0.001	0.000	0.000
$\mu_{\mu_{13}}$	1.421	4.613						
$\sigma_{\mu_{13}}$	0.000	0.000						
$\mu_{\sigma_{13}}$	1.027	1.017						
$\sigma_{\sigma_{13}}$	0.000	0.000						
$\mu_{\mu_{14}}$	0.841	3.076	5.269					
$\sigma_{\mu_{14}}$	0.001	0.001	0.001					
$\mu_{\sigma_{14}}$	0.779	0.631	0.731					
$\sigma_{\sigma_{14}}$	0.000	0.000	0.000					
$\mu_{\mu_{15}}$	-0.176	1.143	2.409	3.589	4.751	5.981		
$\sigma_{\mu_{15}}$	0.000	0.000	0.000	0.001	0.001	0.000		
$\mu_{\sigma_{15}}$	0.543	0.372	0.352	0.331	0.332	0.495		
$\sigma_{\sigma_{15}}$	0.000	0.000	0.000	0.001	0.001	0.000		
$\mu_{\mu_{16}}$	-0.441	0.689	1.578	2.500	3.396	4.301	5.241	6.284
$\sigma_{\mu_{16}}$	0.068	0.091	0.124	0.160	0.201	0.224	0.193	0.152
$\mu_{\sigma_{16}}$	0.502	0.276	0.261	0.257	0.260	0.269	0.284	0.441
$\sigma_{\sigma_{16}}$	0.008	0.008	0.011	0.008	0.012	0.005	0.008	0.026
$\mu_{\mu_{17}}$	-0.046	1.069	2.036	2.970	4.067			
$\sigma_{\mu_{17}}$	0.018	0.022	0.019	0.011	0.005			
$\mu_{\sigma_{17}}$	0.496	0.296	0.270	0.282	0.483			
$\sigma_{\sigma_{17}}$	0.002	0.001	0.002	0.003	0.001			
$\mu_{\mu_{18}}$	0.235	1.619	2.926	4.272	5.758			
$\sigma_{\mu_{18}}$	0.064	0.081	0.067	0.040	0.026			
$\mu_{\sigma_{18}}$	0.576	0.388	0.378	0.402	0.584			
$\sigma_{\sigma_{18}}$	0.012	0.004	0.009	0.005	0.006			
$\mu_{\mu_{19}}$	0.566	2.385	4.102	5.749	7.464			

$\sigma_{\mu_{19}}$	0.016	0.034	0.055	0.056	0.028					
$\mu_{\sigma_{19}}$	0.678	0.507	0.482	0.474	0.695					
$\sigma_{\sigma_{19}}$	0.005	0.005	0.004	0.006	0.006					
$\mu_{\mu_{20}}$	0.655	2.850	5.021	7.128	9.241					
$\sigma_{\mu_{20}}$	0.011	0.017	0.020	0.025	0.014					
$\mu_{\sigma_{20}}$	0.764	0.643	0.600	0.595	0.743					
$\sigma_{\sigma_{20}}$	0.004	0.001	0.002	0.001	0.005					
$\mu_{\mu_{21}}$	-0.288	0.731	1.473	2.184	2.872	3.555	4.227	4.939	5.726	6.755
$\sigma_{\mu_{21}}$	0.011	0.013	0.008	0.003	0.004	0.004	0.003	0.002	0.001	0.000
$\mu_{\sigma_{21}}$	0.466	0.246	0.208	0.199	0.194	0.192	0.197	0.213	0.252	0.432
$\sigma_{\sigma_{21}}$	0.001	0.001	0.002	0.000	0.001	0.001	0.000	0.000	0.000	0.000
$\mu_{\mu_{22}}$	-0.016	1.316	2.492	3.641	4.772	5.945	7.115	8.285	9.472	10.807
$\sigma_{\mu_{22}}$	0.069	0.095	0.125	0.138	0.128	0.093	0.066	0.035	0.012	0.002
$\mu_{\sigma_{22}}$	0.545	0.351	0.334	0.330	0.333	0.335	0.333	0.337	0.356	0.518
$\sigma_{\sigma_{22}}$	0.014	0.011	0.007	0.005	0.007	0.009	0.010	0.008	0.005	0.000
$\mu_{\mu_{23}}$	0.314	1.970	3.532	5.079	6.659	8.252	9.838	11.475	13.144	14.936
$\sigma_{\mu_{23}}$	0.111	0.188	0.317	0.423	0.498	0.517	0.524	0.498	0.393	0.217
$\mu_{\sigma_{23}}$	0.615	0.455	0.450	0.448	0.463	0.453	0.458	0.476	0.493	0.659
$\sigma_{\sigma_{23}}$	0.033	0.028	0.036	0.026	0.009	0.007	0.002	0.020	0.039	0.067
$\mu_{\mu_{24}}$	0.660	2.896	4.970	7.004	9.012	11.102	13.212	15.321	17.422	19.464
$\sigma_{\mu_{24}}$	0.064	0.133	0.191	0.204	0.173	0.114	0.062	0.025	0.007	0.002
$\mu_{\sigma_{24}}$	0.788	0.618	0.595	0.579	0.590	0.602	0.603	0.609	0.584	0.707
$\sigma_{\sigma_{24}}$	0.023	0.013	0.007	0.003	0.018	0.013	0.016	0.008	0.003	0.001
$\mu_{\mu_{25}}$	0.924	4.013	7.074	10.072	13.092					
$\sigma_{\mu_{25}}$	0.000	0.000	0.002	0.003	0.001					
$\mu_{\sigma_{25}}$	0.876	0.836	0.805	0.824	0.888					
$\sigma_{\sigma_{25}}$	0.000	0.000	0.001	0.000	0.001					
$\mu_{\mu_{26}}$	-0.176	3.311	6.770	10.193	13.868					
$\sigma_{\mu_{26}}$	0.090	0.181	0.236	0.209	0.108					
$\mu_{\sigma_{26}}$	1.286	1.014	0.983	1.024	1.375					
$\sigma_{\sigma_{26}}$	0.027	0.035	0.001	0.017	0.032					
$\mu_{\mu_{27}}$	-1.072	3.151	7.139	11.001	15.274					
$\sigma_{\mu_{27}}$	0.049	0.044	0.020	0.005	0.000					
$\mu_{\sigma_{27}}$	1.740	1.184	1.125	1.163	1.697					
$\sigma_{\sigma_{27}}$	0.009	0.009	0.007	0.003	0.000					
$\mu_{\mu_{28}}$	-0.070	1.927	3.602	5.171	7.388					
$\sigma_{\mu_{28}}$	0.001	0.001	0.001	0.001	0.000					
$\mu_{\sigma_{28}}$	0.809	0.513	0.462	0.509	0.829					
$\sigma_{\sigma_{28}}$	0.000	0.000	0.000	0.000	0.000					
$\mu_{\mu_{29}}$	0.652	1.988	3.348	4.910	6.491					
$\sigma_{\mu_{29}}$	0.078	0.101	0.062	0.036	0.026					

$\mu_{\sigma_{29}}$	0.485	0.394	0.401	0.443	0.557
$\sigma_{\sigma_{29}}$	0.025	0.012	0.007	0.006	0.009
$\mu_{\mu_{30}}$	0.622	1.989	3.373	4.797	6.242
$\sigma_{\mu_{30}}$	0.004	0.005	0.003	0.002	0.001
$\mu_{\sigma_{30}}$	0.524	0.397	0.399	0.407	0.540
$\sigma_{\sigma_{30}}$	0.001	0.000	0.000	0.000	0.000
$\mu_{\mu_{31}}$	0.717	2.159	3.564	4.949	6.346
$\sigma_{\mu_{31}}$	0.000	0.000	0.000	0.000	0.000
$\mu_{\sigma_{31}}$	0.527	0.413	0.402	0.400	0.515
$\sigma_{\sigma_{31}}$	0.000	0.000	0.000	0.000	0.000

Tab. 4: The specific artificial test results for EM.

	Cluster									
Value	0	1	2	3	4	5	6	7	8	9
μ_{μ_1}	0.453	2.575								
σ_{μ_1}	1.551	0.688								
μ_{σ_1}	0.827	0.479								
σ_{σ_1}	0.498	0.453								
μ_{μ_2}	1.360	1.364	2.823							
σ_{μ_2}	0.000	0.000	0.000							
μ_{σ_2}	0.820	1.602	0.354							
σ_{σ_2}	0.000	0.000	0.000							
μ_{μ_3}	-1.766	0.095	0.657	1.313	2.342	5.221				
σ_{μ_3}	0.000	0.000	0.000	0.000	0.000	0.000				
μ_{σ_3}	0.105	0.341	0.027	0.303	0.622	0.064				
σ_{σ_3}	0.000	0.000	0.000	0.000	0.000	0.000				
μ_{μ_4}	-1.791	0.285	0.721	1.235	1.482	1.828	2.404	5.210		
σ_{μ_4}	0.001	0.001	0.006	0.026	0.134	0.318	0.140	0.016		
μ_{σ_4}	0.099	0.425	0.042	0.036	0.020	0.175	0.550	0.067		
σ_{σ_4}	0.000	0.001	0.003	0.004	0.059	0.086	0.181	0.004		
μ_{μ_5}	1.340	2.889								
σ_{μ_5}	0.000	0.000								
μ_{σ_5}	1.135	0.999								
σ_{σ_5}	0.000	0.000								
μ_{μ_6}	0.449	1.729	3.271							
σ_{μ_6}	0.345	0.284	0.441							
μ_{σ_6}	0.314	1.091	0.641							

σ_{σ_6}	0.424	0.711	0.327					
μ_{μ_7}	0.315	1.655	3.010	3.926	4.538	5.468		
σ_{μ_7}	0.326	0.399	0.563	0.694	0.738	0.468		
μ_{σ_7}	0.204	1.151	0.303	0.059	0.133	0.124		
σ_{σ_7}	0.365	0.420	0.098	0.020	0.383	0.018		
μ_{μ_8}	-1.599	1.157	1.343	1.897	2.549	3.546	4.553	5.608
σ_{μ_8}	1.006	0.072	0.041	0.066	0.424	0.597	0.232	0.086
μ_{σ_8}	5.465	0.862	0.135	0.050	0.255	0.303	0.092	0.128
σ_{σ_8}	2.686	0.291	0.363	0.059	0.193	0.218	0.086	0.025
μ_{μ_9}	1.207	3.412						
σ_{μ_9}	0.000	0.000						
μ_{σ_9}	1.063	1.414						
σ_{σ_9}	0.000	0.000						
$\mu_{\mu_{10}}$	1.256	2.630	3.641					
$\sigma_{\mu_{10}}$	0.000	0.000	0.000					
$\mu_{\sigma_{10}}$	1.058	0.255	1.222					
$\sigma_{\sigma_{10}}$	0.000	0.000	0.000					
$\mu_{\mu_{11}}$	-0.059	0.620	1.711	3.128	4.679	6.967		
$\sigma_{\mu_{11}}$	1.297	0.697	0.777	0.616	0.407	0.006		
$\mu_{\sigma_{11}}$	0.335	0.449	0.418	1.378	0.460	0.052		
$\sigma_{\sigma_{11}}$	0.402	0.364	0.379	1.293	0.254	0.002		
$\mu_{\mu_{12}}$	-0.969	0.375	1.004	1.932	2.870	3.914	5.064	6.964
$\sigma_{\mu_{12}}$	1.286	0.417	0.324	0.598	0.969	1.027	0.672	0.014
$\mu_{\sigma_{12}}$	0.017	0.601	0.182	0.278	0.438	0.457	0.285	0.053
$\sigma_{\sigma_{12}}$	0.019	0.198	0.244	0.415	0.387	0.320	0.315	0.003
$\mu_{\mu_{13}}$	1.669	4.619						
$\sigma_{\mu_{13}}$	0.000	0.000						
$\mu_{\sigma_{13}}$	1.566	1.275						
$\sigma_{\sigma_{13}}$	0.000	0.000						
$\mu_{\mu_{14}}$	1.353	3.384	5.143					
$\sigma_{\mu_{14}}$	0.000	0.000	0.000					
$\mu_{\sigma_{14}}$	1.211	0.597	0.787					
$\sigma_{\sigma_{14}}$	0.000	0.000	0.000					
$\mu_{\mu_{15}}$	-3.046	0.092	1.671	2.922	3.928	5.180		
$\sigma_{\mu_{15}}$	0.000	0.877	0.639	0.444	0.645	0.188		
$\mu_{\sigma_{15}}$	1.000	0.599	0.602	0.330	0.345	0.718		
$\sigma_{\sigma_{15}}$	0.000	0.428	0.415	0.489	0.358	0.244		
$\mu_{\mu_{16}}$	-2.748	-0.089	0.846	2.124	3.178	4.013	4.752	5.725
$\sigma_{\mu_{16}}$	0.688	0.848	0.966	0.765	0.689	0.764	0.863	0.703
$\mu_{\sigma_{16}}$	0.834	0.655	0.413	0.321	0.366	1.111	3.691	0.353
$\sigma_{\sigma_{16}}$	0.340	0.542	0.407	0.321	0.391	2.150	3.772	0.357

$\mu_{\mu_{17}}$	0.589	1.631	2.233	2.972	3.434					
$\sigma_{\mu_{17}}$	0.094	0.286	0.269	0.247	0.847					
$\mu_{\sigma_{17}}$	0.190	1.233	0.252	0.893	0.335					
$\sigma_{\sigma_{17}}$	0.216	0.272	0.449	0.297	0.494					
$\mu_{\mu_{18}}$	0.216	1.203	2.516	4.066	5.444					
$\sigma_{\mu_{18}}$	0.000	0.000	0.000	0.000	0.001					
$\mu_{\sigma_{18}}$	0.616	0.335	0.489	0.656	0.751					
$\sigma_{\sigma_{18}}$	0.000	0.000	0.000	0.001	0.000					
$\mu_{\mu_{19}}$	-0.483	1.171	3.522	6.345	7.053					
$\sigma_{\mu_{19}}$	0.000	0.000	0.000	0.000	0.000					
$\mu_{\sigma_{19}}$	0.292	0.906	2.083	1.634	0.055					
$\sigma_{\sigma_{19}}$	0.000	0.000	0.000	0.000	0.000					
$\mu_{\mu_{20}}$	0.665	2.889	5.357	6.875	8.667					
$\sigma_{\mu_{20}}$	0.355	0.928	0.278	0.006	0.068					
$\mu_{\sigma_{20}}$	0.775	2.040	1.427	0.185	1.175					
$\sigma_{\sigma_{20}}$	0.341	0.405	1.412	0.052	0.047					
$\mu_{\mu_{21}}$	-2.536	1.009	1.310	2.208	2.845	3.557	4.582	5.331	6.420	7.111
$\sigma_{\mu_{21}}$	0.000	0.014	0.002	0.024	0.021	0.081	0.528	0.272	0.211	0.146
$\mu_{\sigma_{21}}$	0.068	0.836	0.162	0.117	0.029	0.791	0.090	0.463	0.095	0.249
$\sigma_{\sigma_{21}}$	0.000	0.006	0.003	0.018	0.007	0.395	0.119	0.022	0.167	0.037
$\mu_{\mu_{22}}$	0.378	1.795	2.639	3.933	4.875	6.213	6.940	8.165	9.843	11.245
$\sigma_{\mu_{22}}$	0.877	1.019	0.876	0.651	0.788	0.459	0.644	0.933	0.679	0.513
$\mu_{\sigma_{22}}$	0.657	0.758	0.484	1.154	0.652	0.155	0.911	1.022	0.758	0.314
$\sigma_{\sigma_{22}}$	0.460	0.317	0.658	1.127	0.770	0.146	1.117	0.730	0.313	0.188
$\mu_{\mu_{23}}$	0.040	1.799	2.154	4.235	5.767	7.385	10.153	12.122	13.740	15.198
$\sigma_{\mu_{23}}$	0.001	0.072	0.000	0.089	0.954	1.568	1.910	1.088	0.474	0.367
$\mu_{\sigma_{23}}$	0.105	1.538	0.004	1.260	0.416	3.430	2.614	1.092	0.876	0.425
$\sigma_{\sigma_{23}}$	0.003	0.045	0.000	0.507	1.235	1.985	1.240	0.988	0.576	0.305
$\mu_{\mu_{24}}$	0.832	3.036	5.148	6.759	10.258	11.546	13.461	15.889	17.787	19.421
$\sigma_{\mu_{24}}$	0.328	0.540	0.600	1.117	0.829	0.781	0.593	0.745	0.090	0.146
$\mu_{\sigma_{24}}$	0.973	1.172	0.424	4.726	3.803	0.253	1.382	2.054	0.721	0.715
$\sigma_{\sigma_{24}}$	0.288	0.177	1.140	1.704	2.793	0.755	1.090	1.161	0.504	0.092
$\mu_{\mu_{25}}$	0.889	4.011	6.864	10.010	13.267					
$\sigma_{\mu_{25}}$	0.000	0.000	0.000	0.000	0.000					
$\mu_{\sigma_{25}}$	0.841	1.377	0.525	3.864	0.676					
$\sigma_{\sigma_{25}}$	0.000	0.000	0.000	0.000	0.000					
$\mu_{\mu_{26}}$	-0.055	2.209	3.510	8.019	12.834					
$\sigma_{\mu_{26}}$	0.496	0.106	0.454	0.206	0.055					
$\mu_{\sigma_{26}}$	2.757	0.510	5.217	7.093	4.257					
$\sigma_{\sigma_{26}}$	0.481	1.251	1.733	0.045	0.066					
$\mu_{\mu_{27}}$	0.594	2.661	5.610	9.593	14.475					

$\sigma_{\mu_{27}}$	1.859	2.125	3.099	3.312	3.749
$\mu_{\sigma_{27}}$	2.249	8.751	3.432	4.739	7.184
$\sigma_{\sigma_{27}}$	3.570	1.704	3.683	1.576	3.951
$\mu_{\mu_{28}}$	1.275	3.544	4.631	6.056	6.657
$\sigma_{\mu_{28}}$	0.029	0.508	0.448	1.041	0.575
$\mu_{\sigma_{28}}$	2.083	1.210	0.394	0.295	2.077
$\sigma_{\sigma_{28}}$	0.033	0.361	0.147	0.108	0.099
$\mu_{\mu_{29}}$	0.801	2.235	3.432	5.096	6.892
$\sigma_{\mu_{29}}$	0.115	0.274	0.558	0.381	0.052
$\mu_{\sigma_{29}}$	0.417	0.758	0.130	0.844	0.258
$\sigma_{\sigma_{29}}$	0.051	0.014	0.101	0.266	0.030
$\mu_{\mu_{30}}$	0.185	1.489	3.546	4.947	5.831
$\sigma_{\mu_{30}}$	0.000	0.000	0.000	0.000	0.000
$\mu_{\sigma_{30}}$	0.349	0.627	1.369	0.018	0.642
$\sigma_{\sigma_{30}}$	0.000	0.000	0.000	0.000	0.000
$\mu_{\mu_{31}}$	0.423	1.292	2.994	4.988	6.189
$\sigma_{\mu_{31}}$	0.167	0.397	0.606	0.324	0.134
$\mu_{\sigma_{31}}$	0.202	0.671	1.337	0.795	0.512
$\sigma_{\sigma_{31}}$	0.098	0.209	0.182	0.112	0.040